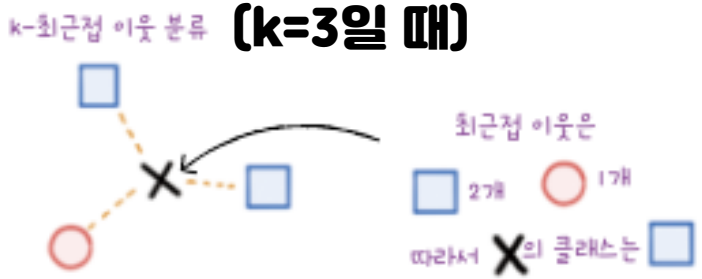
머신러닝 & 딥러닝 2

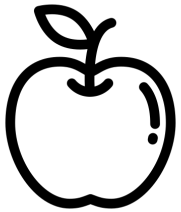
AI 학술동아리 <MLP>

- Index

1-1. Supervised Learning(지도 학습) - Classification(분류)



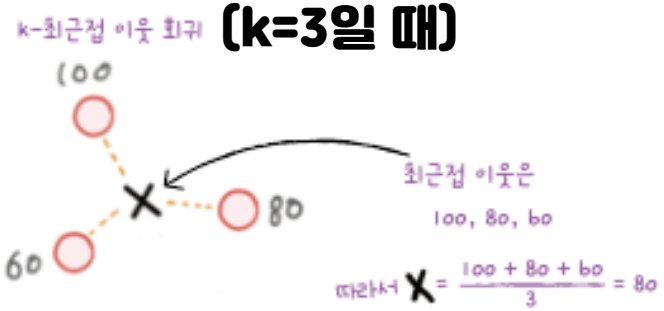
(k=3일 때)



- training data(훈련 데이터)
  - input(입력)
    - 길이 feature(특성)
      - 길이 : 10cm
      - 길이 : 20cm
    - 무게 feature(특성)
      - 무게 : 300g
      - 무게 : 200g
  - target(정답) /label
    - 사과 : 0
    - 바나나 : 1

여러 사과(바나나) 데이터 중에 하나의 사과(바나나) 데이터 - sample(샘플)

1-2. Supervised Learning(지도 학습) - Regression(회귀)



(k=3일 때)

training data(훈련 데이터)
- input(입력)
- target(정답) /label

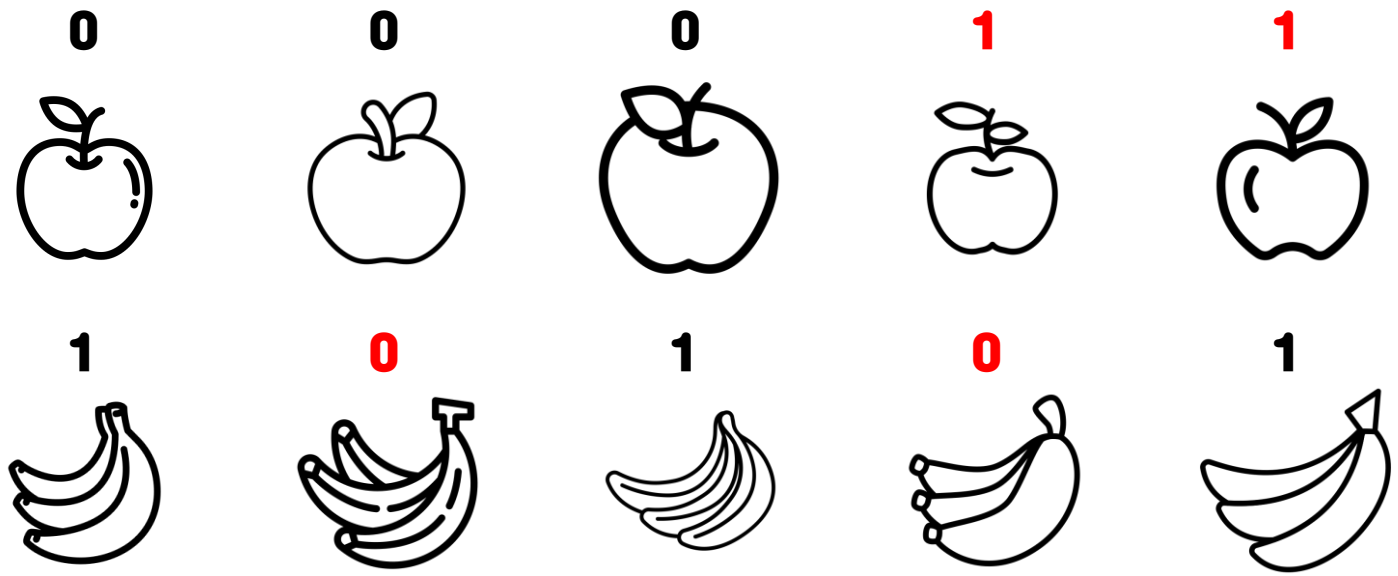
길이 feature(특성)

길이 :	10cm	12cm	13cm
무게 :	300g	330g	340g

길이 : 12.5cm, 무게는? (300 + 330 + 340) / 3 = 323.3g

2-1. Accuracy(정확도)

사과 : 0, 바나나 : 1



10개 중 6개만 정답!
accuracy: 6/10 = 0.6

2-2. Coefficient of determination (결정계수) = R^2

$$R^2 = 1 - \frac{(\text{타깃} - \text{예측})^2\text{의 합}}{(\text{타깃} - \text{평균})^2\text{의 합}}$$

쉽게 말해
정답값과 예측값의 차이가 적을 수록(비슷하게 맞출수록) 분자는 0에 가까워짐 -> R^2이 1에 가까워짐
정답값과 예측값의 차이가 클 수록(잘 못맞출수록) 분자는 커짐 -> R^2이 0에 가까워짐

3. Mean Absolute Error(평균절대오차)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

$$y_i : target, \hat{y}_i : prediction$$

평균적으로 예측값이 정답값과 얼마나 차이가 나는지 알 수 있음

4-1. Overfitting(과대적합)

Train Set Score: 0.96
Test Set Score : 0.88

train set에서 점수가 좋았지만, test set에서 점수가 나쁜 경우 -> overfitting
=> 새로운 sample에 대한 예측이 잘 맞지 않을 것임

4-2. Underfitting(과소적합)

Train Set Score: 0.96
Test Set Score : 0.99

or

Train Set Score: 0.76
Test Set Score : 0.73

train set보다 test set의 점수가 높거나, 두 점수가 모두 너무 낮은 경우 -> underfitting
- 모델이 너무 단순하여 train set에 적절히 훈련되지 않음

과소적합의 또 다른 원인은 train set과 test set의 크기가 매우 작기 때문

k-NN 알고리즘 단점

- k-NN 알고리즘 : 데이터를 모두 가지고 있는 것이 전부

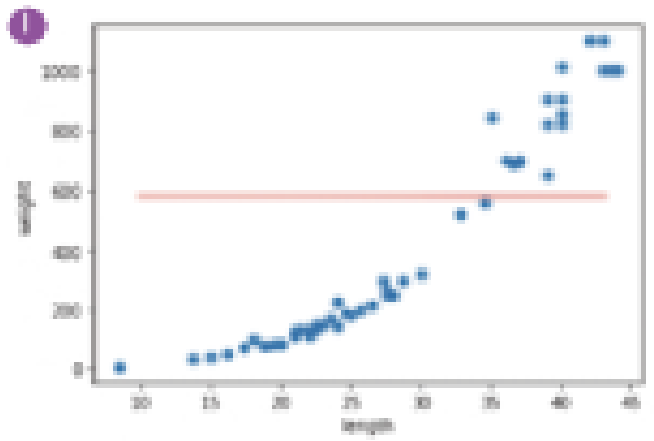
  => 데이터가 아주 많은 경우 사용이 어려움

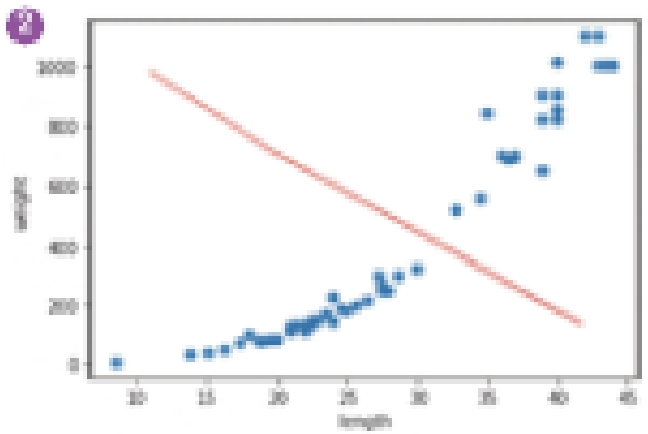
  => train set 범위 밖의 샘플을 예측할 수 없음

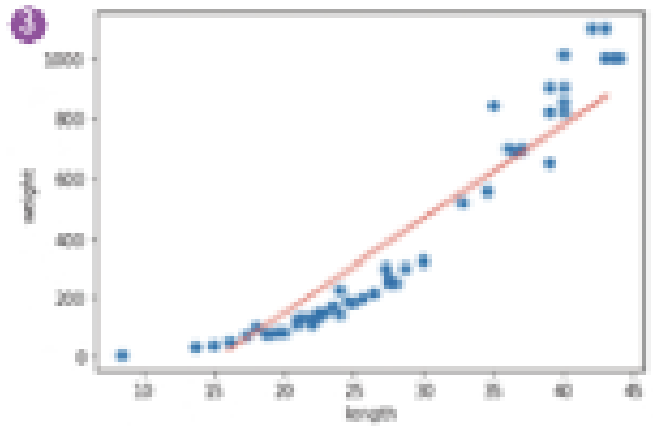
5. Linear Regression(선형 회귀) 알고리즘

feature를 **가장 잘 나타낼** 수 있는 **선**을 학습하는 알고리즘

target = a * feature + b

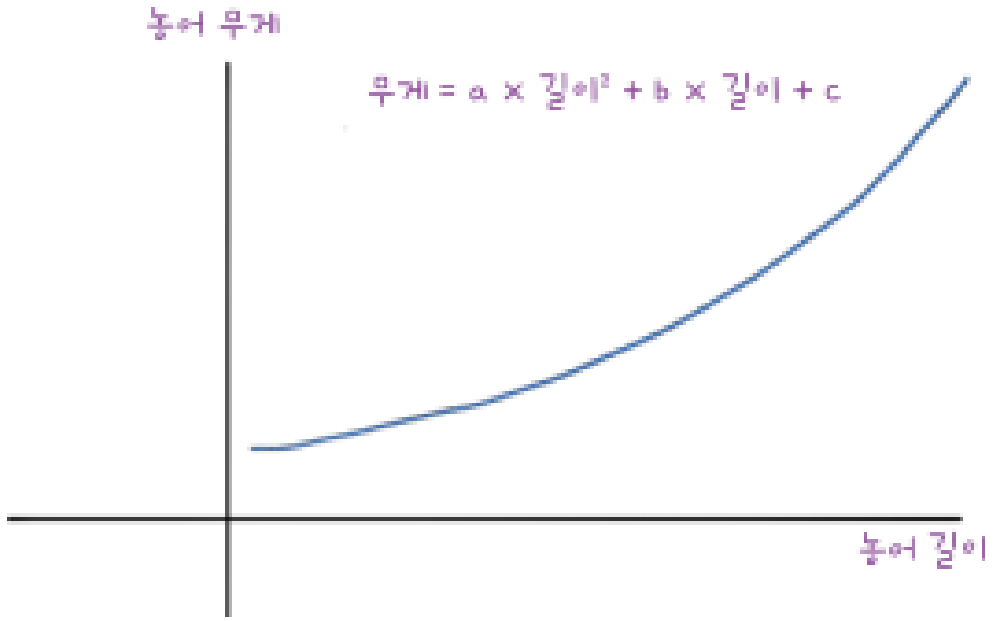






5-1. Polynomial Regression(다항 회귀) 알고리즘

feature를 **가장 잘 나타낼** 수 있는 **곡선**을 학습하는 알고리즘



다항 회귀도 보통 **Linear Regression으로 표현**함

Linear Regression ⊃ Polynomial Regression

Tip! Parameter

- 사람이 정해주는 parameter :
  hyperparameter(하이퍼파라미터)
  - 예) k-NN 알고리즘의 k값

- 모델이 feature에서 학습한 parameter :
  model parameter(모델 파라미터)
  - 예) Linear Regression 알고리즘의 coef_와 intercept_ 값

Tip! Based Learning(기반 학습)

- Train Set를 저장하는 것이 train의 전부 :

  Instance-Based Learning(사례 기반 학습)

  - 예) k-NN 알고리즘(모델 파라미터 X)

- 최적의(optimal) 모델 파라미터를 찾는 것 :

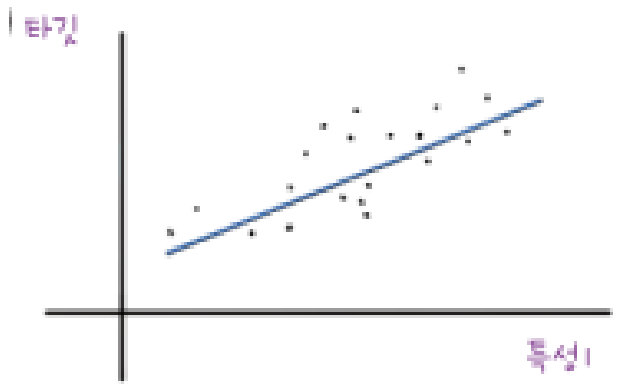
  Model-Based Learning(모델 기반 학습)

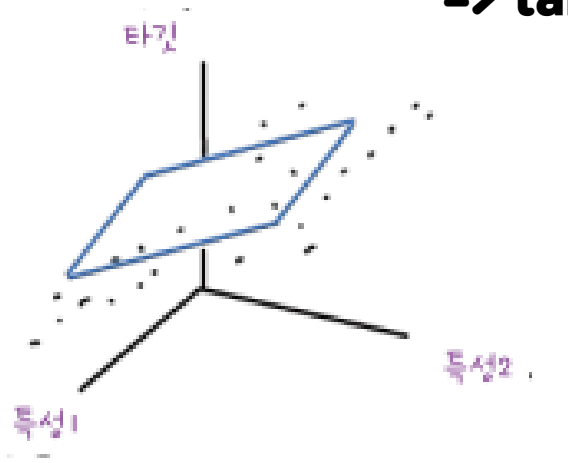
  - 예) Linear Regression 알고리즘에서 적절한 coef_와 intercept_ 값을 찾기

6. Multiple Regression(다중 회귀)

여러 개의 feature를 사용한 Linear Regression

feature가 많은 고차원에서는 Linear Regression이 매우 복잡한 모델을 표현 가능





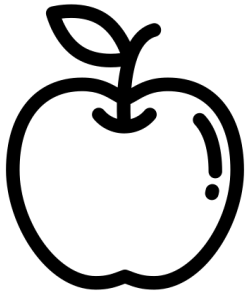
feature 2개 일 때
=> target = a * feature1 + b * feature2 + c

feature 개수가 늘어날 때마다 학습하는 계수가 하나씩 증가

7. Feature Engineering(특성 공학)

기존 feature를 사용해 새로운 feature를 뽑아내는 작업 <- 복잡한 모델을 표현하기 위함

feature 개수를 크게 늘리면 선형 모델은 train set에 대해 거의 완벽하게 학습할 수 있음
하지만, 이런 모델은 train set에 너무 overfitting되어 test set에서 점수가 좋지 않음



길이: 10cm

둘레: 25cm

길이의 제곱 : 100

길이 x 둘레 : 250

둘레의 제곱 : 625

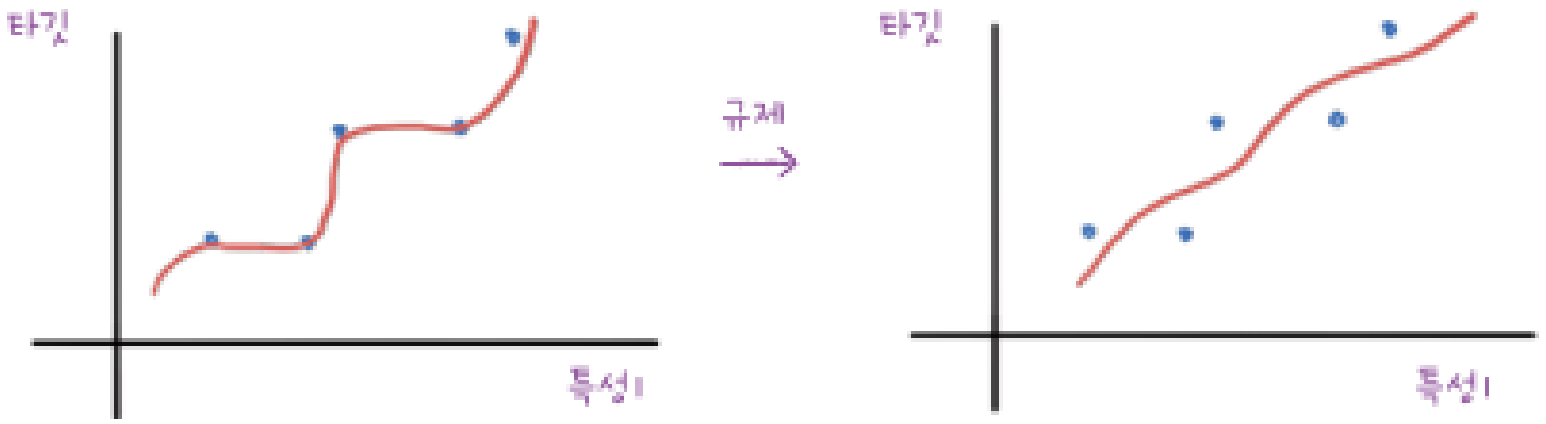
Tip! transformer, estimator

- **transformer(변환기)** – 특성을 만들거나 전처리하기 위한 클래스
  - 예) PolynomialFeatures 클래스
- **estimator(추정기)** – 모델 클래스
  - 예) LinearRegression 클래스

8. Regularization(규제)

모델이 train set를 너무 과도하게 학습하지 못하도록 막는 것 = overfitting 되지 않도록 만드는 것
Linear Regression의 경우 feature에 곱해지는 계수(또는 기울기)의 크기를 작게 만듦

중요! feature의 scale은 모두 다르기(쉽게, 단위가 다르기) 때문에
normalization(정규화) 과정을 반드시 거치고 regularization(규제)를 적용해야 함



Normalization -> Regularization

Remind. Normalization

- Standard Score(표준점수) = Z-Score
  - 각 feature값이 mean에서 std의 몇 배만큼 떨어져 있는지 나타냄

- $z = \frac{x-\mu}{\sigma}$, μ : mean(평균), σ : std(표준편차)

- 중요! 반드시 train set의 mean과 std를 이용해서 test set를 normalization 해야 함

8-1. Ridge(릿지), Lasso(라쏘)

선호됨

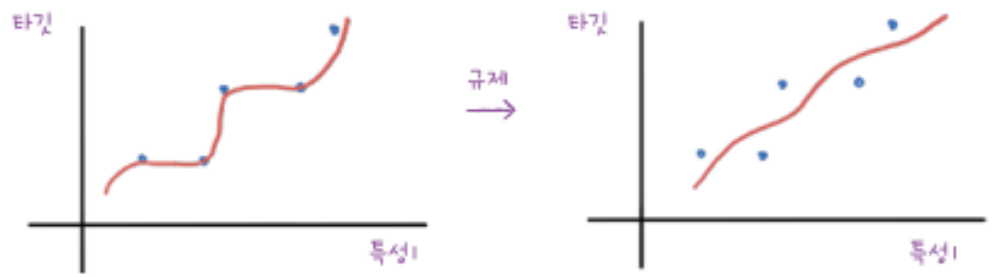
Linear Regression Model + Regularization

- **Ridge : 계수를 제곱한 값을 기준으로 규제**
- **Lasso : 계수의 절댓값을 기준으로 규제**
  - Lasso 모델은 계수 값을 0으로 만들 수 있음
  - -> 해당 feature가 쓰이지 않은 것과 같음

alpha 매개변수 값으로 regularization의 강도를 조절(alpha값이 클 수록 규제 강도가 세짐)

- alpha값이 크면, 계수 값을 더 줄이고 조금 더 underfitting 되도록 만듦
- alpha값이 작으면, 계수를 줄이는 역할이 줄어들고 Linear Regression 모델과 유사해져 overfitting 가능성이 큼

요약



alpha값이 작을 때 alpha값이 클 때

Tip! scikit-learn 모델 사용법 간단 정리

```
# k-NN 알고리즘 사용 (기본 k값은 5)
from sklearn.neighbors import KNeighborsClassifier

# 모델 생성
kn = KNeighborsClassifier()

# 모델 training
kn.fit(train_input, train_target)

# 모델 평가 (0 ~ 1사이의 값을 반환, 1에 가까울수록 모델의 성능이 좋음을 나타냄)
kn.score(test_input, test_target)

# 정답 예측
kn.predict(test_input)
```